JACG NEWSLETTER
Vol.5 No.2
Oct. 1985
Single Copy
Price $1.50

THE JERSEY ATARI COMPUTER GROUP

884-1642 JACG HOTLINE 884-1642

From the Editor's Desk...

Election day will soon be upon us. We have several people who have come forward and volunteered their time to serve us as officers. This is a sign that the J.A.C.G. is alive and well.

If you decide between now and the November meeting that you would like to become a candidate please contact Art Leyenberger or be prepared to have your name placed in nomination from the floor. We encourage and welcome all candidates.

If you are not going to be a candidate it still is important that you participate by casting your votes for those persons you think will serve you best. Plan on exercising that privilege and responsibility next month. The leaders you choose will directly affect your club activities for the next twelve months. Don't treat your vote lightly.

Frank Pazel

Frank Pazel
Editor-in-Chief, JACG Newsletter

In This Issue

From The Conn - A. Leyenberger......................2
Spreadsheets For The Mathematician - D. Forbes......3
Why Arrays? - R. Kushner............................4
Voters In ACTION! - C. Lichtnewalner................6
Halloween Pictures From J.A.C.G. Disk...............8
September Meeting Highlights - J. Kennedy...........8
Forth In The Laboratory - D. Forbes.................13
Chain Letters - K. Pietrucha........................15
Peeks And Pokes - K. Pietrucha......................16
Cartoons By Tony Pellechio..........................6,15

NEW J.A.C.G. PRINT SHOP PICTURES!

SEE PAGE 8

HAPPY HALLOWEEN

MARK YOUR CALENDARS!!
JACG Meeting Schedule

November 9, 1985
December 14, 1985
January 11, 1986
February 8, 1986
March 8, 1986
April 12, 1986

From the Conn.....

This is being written on an Atari 520ST using AtariWriter. Well, that's not quite true. ST-Writer is the program I am currently using on the ST but it is an exact clone of AtariWriter. It sure is nice to be using a familiar program (that works) on the ST computer.

The folks at Atari have recently released two public domain programs for the ST: ST-Writer and NEO-Chrome. NEO is a painting program, the one that was used to create the beautiful waterfalls picture that many of you have already seen demoed on the ST computer with an RGB color monitor.

At the September meeting, I asked for volunteers and/or nominations to fill the club posts that will become vacant in November. I was less than thrilled with the response I got. I figured it was the usual member apathy rearing its ugly head again. But wait, I was in for quite a pleasant surprise. In the couple of weeks following the meeting, I have been contacted by several people who have come forward to run/volunteer for office. Here is how the ballot stands:

President: Nick Scalera, Bill Martin
Vice Pres.: Scott Brause
Secretary: Bob Mulhearne
Treasurer: Shree Vandenberg
Librarian: Don Ursem
Editor: Frank Pazel
Program Chair: Jerry Frese
BBS SYSOP: Scott Brause
Advertising: Helene Rotondo

So we have a full slate of officers for the coming year and need only to vote on the president of the club at our November meeting. Of course, according to the official bylaws, the October meeting is open to additional nominations and/or volunteers.

Another topic of concern to all members of the JACG is how we will handle the new ST computer. Since 95 percent of our members have 8-bit Atari computers we will not forsake them at all. However, 8-bit computer technology is dead or at least in its final death throws so we will have to incorporate the new 16 bit computers into our activities.

What I propose is that starting in November, a portion of the meeting (at the end) be set aside for ST activities. I'd guess from 11:45 to 12:15 would be appropriate. In this segment we can cover ST demos, news and any other topics of interest to ST owners. However, I believe that if an ST-related topic is of general interest to the members than it should be covered during the main body of the meeting. The Program Chairman will decide on the appropriateness of topics.

If there is no objection, I plan to be the ST coordinator for the club. I will MC the ST portion of the meetings and maintain the ST disk library (did you know we now have 12 volumes in the ST library). If you have any questions, suggestions or information you would like to share about the ST computer, please contact me.

The club has just purchased an Atari 520ST computer system. It consists of the 520ST computer, one single-sided disk drive and an RGB color monitor. We will also be buying a carrying case for the system shortly. For the present, we will have to use a video camera to get the ST screen on our projection TV since there is no RF or standard video output on the ST. This shouldn't pose any problem other than me having to schlep a video camera to each meeting. Eventually, we will be getting an adapter to go from the ST to a standard TV signal. I can't tell you any more about this at this time.

Arthur Leyenberger
President, Jersey Atari Computer Group

BULLETIN BOARD

24 HOURS/DAY

201-549-7591

DAVID E. MARCUS
BOX 3130
EL CENTRO CA 92244-3130
September 10, 1985

Frank Pazel
Editor-in-Chief
JACG NEWSLETTER
14 WHITMAN DR
DENVILLE NJ 07834

JACG NEWSLETTER VOL 5 No 1 9/85

I am the not so proud owner of a new 130XE, having given my 1200XL to my son who is now away at college. The 130XE does NOT run everything that the previous models ran. It will not run Print Shop or Dimension X no matter what you do. Broderbund says they know all about it--"too bad, so sad." I was, therefore, somewhat distressed to see that you printed Neil Harris'es misrepresentations in full on page 2. The second misrepresentation was that you can dump something from the RAM disk to some other disk drive. It may work with some terminal programs, but it sure doesn't work with GT DATA MANAGER--the data base that comes with the Indus GT. The program loads into RAM just fine, and runs super fast. It works just great--until you want to SAVE what you've done. Good luck. In order to do this, you have to exit to DOS. And, when you do, the DOS re boots. Mem.sav has no effect, and all your data is lost. Nice design.

Also, you'd think that inserting a cartridge would disable BASIC. You'd be wrong. If you don't hold down OPTION, at least with ATARIWRITER, the printer goes crazy after the program loads.

With reference to incompatibility, I've tried both sides of the translator, and holding down OPTION when the computer coughs and starts to reboot--the whole ball of wax. No go. ANTIC has a disk they call the "The FixXL (PD)" and add "000 OS overlay for XL & XE" which they advertise as the ultimate answer to the incompatibility problem. It's not. By now everyone has a copy of this public domain program either from a BBS or from a user group, and ANTIC's failure to adequately describe it so that we could save the $10 to try on something we don't have that might work, amounts to false advertising. From their ad, you would have thought that this was something NEW. It's not.

Kindest regards,

DAVID E. MARCUS

cc: ATARI EXPLORER
JACE
ANTIC
Carl M. Evans

P.S. I ENCLOSE A S.A.S.E., PLEASE ASK ONE OF THE LIBRARIANS TO SEND ME A LIST OF AVAILABLE DISKS. THANKS.

Spreadsheets for the Mathematician

by Donald Forbes - JACG

Ever think of using a spreadsheet package, such as Visicalc, to do numerical analysis? Or research mathematics? Or teaching math at the college or graduate level? Crazy??? Believe me, it works!

Albert Einstein said that "every physicist should have a shoemaker's job." So if your shoemaker's job is doing spreadsheets (today's equivalent of economics, which used to be called the 'dismal science') when you would rather be doing mathematical research, then rejoice! You have been liberated at last.

The idea is stupidly simple -- once someone points it out to you. (I am assuming that you know how a spreadsheet works.) Suppose you want to compute a Fibonacci series which goes 1,1,2,3,5,8,... where each number is the sum of its two predecessors.

All you have to do is set up your formulas like this:

Formulas

	A	B
1	FIB NOs	RATIO
2	1	
3	1	+A3/A2
4	+A2+A3	+A4/A3
5	+A3+A4	+A5/A4
6	+A4+A5	+A6/A5
	Replicate	Replicate
22	+A20+A21	+A22/A21

Then you get your output (including the convergent ratio) like this:

Output

	A	B
1	FIB NOs	RATIO
2	1	
3	1	1
4	2	2
5	3	1.5
6	5	1.666667
22	2584	1.618034

That's the whole idea! Nothing could be simpler. To compute factorials you use column A as a counter and in column B you multiply by the factorial you computed in the preceding row of column B.

Want to do linear regression? Put your string of x values in column A, your string of y values in column B, x squared in column C, y squared in column D, the x and y cross products in column E and the computed regression line (MX+B) in column G. Could anything be easier?

This is just the beginning. Here is a list of the goodies: Fibonacci numbers, factorials, bisection algorithm, fixed-point algorithm, limits, differentiation, Newton's method, numerical integration, Taylor polynomials, differential equations, polynomial interpolation, matrix multiplication, Jacobi and Gauss-Seidel algorithms, dominant eigenvalue, eigenvalues and diagonalization, continued fractions, Euclid's GCD algorithm, binomial theorem, synthetic division, contour graphs, modular arithmetic, Russian peasant multiplication, statistics (mean, correlation, regression, confidence intervals), probability (Bayes' rule), algebra word problems, trigonometry, compound interest, personal finance model, simplex linear programming, game theory, matrix powers, systems of linear equations, boundary-value problems.

Where is this treasure trove? In a new $17 165-page McGraw-Hill paperback called Mathematical Applications of Electronic Spreadsheets by professor Deane E. Arganbright of the Whitworth College department of math and computer science in Spokane WA. His phone number is 509/466-1000 x474 if you want to call him.

He points out that any "spreadsheet program calculates values for each of the expressions and displays the data and the calculated values in a spreadsheet format on the computer screen. By changing the entries in the matrix a user can modify the hypotheses or parameters of the model..."

"The electronic spreadsheet can also be used effectively and creatively in mathematics. Algorithms that are recursive, iterative or adaptable to a table format can often be implemented easily and naturally on a spreadsheet, allowing the user to change initial values, step sizes, and other parameters and see the result of the changes instantly. Moreover, many applied mathematics problems can also be set up, analyzed, and solved on a spreadsheet. The what-if features of the spreadsheet make it useful in mathematical modeling, the design and study of algorithms, problem solving, and the study of mathematics."

The book is written cookbook style: for each of the 36 topics there is an introduction, followed by SPREADSHEET CONSTRUCTION, then USER INTERACTION, then EXERCISES and MODIFICATIONS, and finally REFERENCES. He assumes you know the math and makes no effort to teach it, but does give you 65 textbook references with page numbers, and a dozen references to the workings of spreadsheets. His demos are done in Visicalc, but will work with Multiplan or Framework or any of the other popular packages.

Arganbright sums it all up as follows: "There are a number of reasons why the electronic spreadsheet is an especially good means of integrating the computer into the study of mathematics:

"Spreadsheet operation is natural and easy to learn... Spreadsheets for algorithms follow the same format and techniques commonly used for doing the work by hand... The spreadsheet format makes algorithms and manipulations concrete and easy to visualize... The what-if capabilities of a spreadsheet program allow a user to modify parameters easily and see the effects of changes instantly... Learning how to use a spreadsheet program provides experience and skills that are valued in the business community... Moreover the range of possible applications seems unending..."

"Of course, there is another fundamental reason for doing mathematics on a spreadsheet: It's fun!"

Why Arrays?

by Richard Kushner

You're the coach of a Little League team with 25 kids on your team. You've just given out the uniforms and have a list of who has which uniform number. You've also recently purchased an Atari 800XL computer and disk drive (or tape recorder) and you want to use the Atari to keep this list. You figure that this will make it easy to make changes if numbers change or if you add any new players or if anyone changes teams. You've studied a little of Atari BASIC but aren't sure what the best way is to proceed.

Undaunted, you set out to write a program. You start by defining each of the players as a separate numeric variable, such as PLAYER1, PLAYER2, PLAYER3,..., up to PLAYER25. You already know that a "numeric variable" is just something that gets assigned a number value in your program. This value may or may not change in the program. The idea of thinking of a variable as a pigeonhole, a place to store things, is clear to you. You also know that in Atari BASIC variable names can be as long as a program line, must begin with a capital letter and can only use capital letters and numbers with no spaces. You've read that it is good practice to use names that provide some information about what they stand for. Thus, PLAYER1, is much better than P1 or PL1. You know you're not the world's greatest typist, so you want to keep away from overly long variable names like LITTLELEAGUEPLAYER1.

You want your program to ask you to tell it the players' uniform numbers and you want it to be able to list out all the players and their numbers on the screen. Scratching your head, you begin.

The first part of your program looks like Listing 1.

LISTING 1

```
10 PRINT "INPUT PLAYER1 UNIFORM NUMBER"
20 INPUT PLAYER 1
30 PRINT "INPUT PLAYER2 UNIFORM NUMBER"
40 INPUT PLAYER 2
50 PRINT "INPUT PLAYER3 UNIFORM NUMBER"
60 INPUT PLAYER 3
70 PRINT "INPUT PLAYER4 UNIFORM NUMBER"
80 INPUT PLAYER 4
.
. INPUT FOR PLAYERS 5 TO 23 HERE
.
480 PRINT "INPUT PLAYER24 UNIFORM NUMBER"
490 INPUT PLAYER 24
500 PRINT "INPUT PLAYER25 UNIFORM NUMBER"
510 INPUT PLAYER 25
```

It seems a little troublesome to have to use all that programming (and typing!) to do what seems like a pretty simple job. Oh well, you shrug your shoulders and continue onward.

The screen display part of the program looks like Listing 2.

LISTING 2

```
1000 PRINT "PLAYER1 HAS UNIFORM NUMBER "
;PLAYER1
1010 PRINT "PLAYER2 HAS UNIFORM NUMBER "
;PLAYER2
1030 PRINT "PLAYER3 HAS UNIFORM NUMBER "
;PLAYER3
1040 PRINT "PLAYER4 HAS UNIFORM NUMBER "
;PLAYER4
.
.LISTING OF UNIFORMS FOR PLAYERS 5 TO 23 HERE
.
.
1240 PRINT "PLAYER24 HAS UNIFORM NUMBER "
;PLAYER24
1250 PRINT "PLAYER25 HAS UNIFORM NUMBER "
;PLAYER25
```

Now things are really troubling. Again you needed a separate program line for each player. What if the whole league of 250 players wanted you to do a similar list? There is a growing belief that there must be a better way!

This example is perhaps a little simple minded, but it does illustrate some important points regarding the use of variables in Atari BASIC programs. Giving each variable in a similar group different names is important to distinguish them from each other. However, the approach above makes for lots of typing and many lines of programming. If we wanted to find out who had uniform number 88 we would have to construct a similarly lengthy section of programming. As our Little League coach asked, isn't there a better way?

Yes, there is. Using an *array* is the answer. But first things first. The first question is "what is an array?". One way of describing an array is to say that it permits the programmer to designate a collection of numeric variables with one variable name. Now, instead of designating the uniform numbers as PLAYER1, PLAYER2, PLAYER3, and so forth, you may use an array variable. PLAYER(X) may be used to refer to the uniform number of the Xth player. PLAYER(X) is read "PLAYER sub X". The value in parenthesis is called a *subscript*. Each data value in the array is called an *element*. Using an array we could do the uniform record keeping with a very simple piece of BASIC programming. The power of arrays is that they permit indirect addressing, which is a fancy way of saying that the "X" in PLAYER(X) can be made to vary to fill in or find certain array elements at the program's discretion. The name "PLAYER" is attached to all array elements, only the subscript is unique. This is in sharp contrast to using single variables where you cannot change just one part of the name to select different members of the group.

This is shown in LISTING 3 which accomplishes the same job as LISTING 1 (and is much shorter).

LISTING 3

```
10 DIM PLAYER(25)
20 FOR I=1 TO 25
30 PRINT "INPUT THE JERSEY NUMBER FOR
PLAYER#";I
40 INPUT PLAYER
50 PLAYER(I)=PLAYER
60 NEXT I
```

This is an example of a FOR...NEXT loop. We begin by carrying out an operation known as *dimensioning* the array. This is done in line 10. The DIM statement sets aside space (pigeonholes) for an array to be named PLAYER. It is required that you DIMension an array before you use it. Otherwise you will get an "ERROR 9" message whenever you refer to the array in your program. Lines 20-60 constitute the loop used to fill the array with the uniform numbers. We could have gotten fancier and put in some statements to protect against bad input (such as inputting letters instead of numbers) and we could have permitted the user to stop before all 25 values had been given. However, we wanted to keep this clean and neat in order to concentrate on the array aspects of the program. You can create an equally simple piece of programming to print out the uniform numbers, just as the cumbersome programming in LISTING 2 did. We'll leave that to you to work out. Also, simply by changing the values in lines 10 and 20, you can expand to as large an array as your computer can handle.

Before we go any further, let's state some rules that must be followed when using numeric arrays. Keeping these in mind will prevent all manner of difficulty in your programs.

RULE 1: All arrays must be DIMensioned before they can be used, no matter how many elements the array will contain. Note also that DIM PLAYER(25) actually reserves space for 26 array elements, since the computer prefers to start counting from 0 rather than 1. You are free to use or ignore the zeroth member of an array.

RULE 2: Be sure your program never goes back to the statement that DIMensioned your array. If this should happen, you will also get an "ERROR 9" warning and the program will halt. To satisfy Rules 1 & 2 is is advisable to put your DIM statements right near the beginning of your program and never loop back to these statements.

RULE 3: You cannot READ or INPUT numeric variables directly into an array. This means that the following statement is not permitted:

```
40 INPUT PLAYER(20)
```

Instead, you must take an indirect approach such as that used in LISTING 3:

```
40 INPUT PLAYER
50 PLAYER(20)=PLAYER
```

We have used PLAYER as a temporary variable to help us INPUT date into our array. The same holds for trying to READ data.

RULE 4: Never assume that the elements of an array that you have created are initialized by the computer to be 0 when you DIMension the array. This is a fairly small point, but one that can get you in trouble. The space set aside for your array may contain some "garbage" numbers; that is, numbers other than zero. If it is important in your program that your array contain all zeros (or any particular value), then put the value you want there before you use the array with a loop such as

```
100 I=1 TO 25
110 PLAYER(I)=0
120 NEXT I
```

In LISTING 3 we didn't need to do this since we were going to input 25 values of uniform numbers anyway.

I trust that you are beginning to see the value and power of using arrays when you have a collection of similar variables. Not only does it make programs shorter and easier to write, it permits you to manipulate the numbers in the array much more readily than if they all had separate names. For example, you might want to have a list of the uniform numbers in numeric order rather than player numbers in numeric order, that is, a list that might look like

UNIFORM #1 PLAYER 22
UNIFORM #2 PLAYER 3
UNIFORM #3 PLAYER 11
and so on.

You would need to go through the list of players and find which player has Uniform #1, Uniform#2, etc. With arrays, this is a relatively easy thing to do. Without them it is another long programming task. We'll leave it to you to try out both approaches.

We'll stop here for this month and pick up again in the next issue when we will expand our discussion to two-dimensional arrays, look at some more examples and summarize.

This article is based on the book Basic Atari BASIC, by James S. Coan and Richard Kushner, published by the Hayden Book Company, Hasbrouck Heights, NJ and available at bookstores and computer stores nationwide. It would make a wonderful Christmas or birthday present. *The article is reprinted from "The Atari Explorers" magazine.*

VOTERS IN ACTION!

Charles P. Lichtenwalner - JACG

In the April 1985 <u>Scientific American</u> "Computer Recreations" column an interesting simulation described as a voting game is a suggested programming recreation. A rectangular grid is populated with a random scattering of "voters of two political parties" (or colors, or symbols.) The program then picks one of the voters on the grid and one of their eight nearest neighbors--all picks at random. Then "the voter's political persuasion becomes that of his neighbor, regardless of earlier belief."

The attached listing is an implementation of this simulation written in ACTION!. Having only a green monitor I choose spaces and *'s to denote the two political parties. I decided to use Graphics 1 to give a square grid. As noted by A. K. Dewdney the appeal is watching large blocks of votes developing which migrate around as the two parties struggle for dominance. Eventually the struggle collapses and the screen fills with a single character (party.) If you believe in contrarian voting as my wife does, you might try modifying the program to an antivoting game where the selected voter adopts an opposite opinion to that of his neighbor.

In a follow-up article in the July <u>Scientific American</u> Mr. Dewdney mentions that some people had trouble noting the collapse of the two party system. He mentioned that his program takes the better part of a day to reach this state. It is not mentioned how large a grid he used, but I generally get a collapse within 2 to 5 minutes with the 20x20 grid of Graphics 1. However, ACTION! is fast. I get about 1000 voter changes per second. I put in a print statement to let me know whenever 65000 changes have taken place.

Another suggestion you may want to try is to set up a multi-party (more than two) world. To do this simply change the data in the FIGS array as noted in the comment. As currently configured up to eight parties can be simulated. Whatever your modification, the interesting part is watching the changing display.

To paraphrase the League, "Get out and VOTERS!"

```
;   VOTERS     C. P. LICHTENWALNER 9/11/85
;AFTER A SUGGESTION IN COMPUTER RECREATIONS
;COLUMN OF "SCIENTIFIC AMERICAN" APRIL 1985

BYTE SAVMSCL=88, SAVMSCH=89,TYPRTY
;CHANGE THE FIGS ARRAY TO VARY THE CHARACTER
;PRINTED ON THE SCREEN
;OR BIAS THE INITIAL VOTER DISTRIBUTION
;OR TO SET UP A MULTI-PARTY SYSTEM
BYTE ARRAY FIGS=[Ø Ø Ø Ø 1Ø 1Ø 1Ø 1Ø], J
INT ARRAY NGHBRS=[65515 65516 65517 65535 1 19 2Ø 21],
NGHBR, CELL
CARD SCREEN,I,II,SCRNLOC

PROC INITGR()
;SET GRAPHICS MODE 1 AND FIND START OF
;SCREEN MEMORY
    GRAPHICS(1)
    SCREEN=SAVMSCL+256*SAVMSCH
RETURN

PROC LDVTRS()
;LOAD THE SCREEN WITH A VALUE FROM FIGS
    INITGR()
    FOR I=Ø TO 4ØØ
        DO
            J=FIGS(RAND(8))
            POKE(SCREEN+I,J)
        OD
RETURN

INT FUNC ADJNGHBR(INT CELL, NGHBR)
;ADJUST VALUE SO IT FALLS BETWEEN Ø-399
;i.e. ACCOUNT FRO WRAPAROUND
       IF (CELL+NGHBR)<Ø THEN RETURN (CELL+NGHBR+4ØØ)
              ELSEIF (CELL+NGHBR)>=4ØØ THEN RETURN
(CELL+NGHBR-4ØØ)
              ELSE RETURN (CELL+NGHBR)
       FI

INT FUNC PICK()
;PICK A RANDOM NUMBER BETWEEN Ø-399
       CARD RANDOM
       BYTE RND
       DO
       RANDOM=PEEK(5377Ø)
       RANDOM=(RANDOM LSH 1)!PEEK(5377Ø)
       UNTIL RANDOM<4ØØ
       OD
RETURN (RANDOM)

PROC VOTERS()
       BYTE KEY
       CARD SCNPC,SCNPN
       LDVTRS()
       FOR II=Ø TO 1ØØ
       DO
              PRINTF("65ØØØ * %U VOTER CHANGES%E",II)
       FOR I=Ø TO 65ØØØ
       DO

;PICK A CELL AT RANDOM THEN A NEIGHBOR
;AT RANDOM AND CHANGE THE CELL TO MATCH
;ITS NEIGHBOR
                CELL=PICK()
                NGHBR=NGHBRS(RAND(8))
                SCRNLOC=ADJNGHBR(CELL,NGHBR)+SCREEN
                TYPRTY=PEEK(SCRNLOC)
                POKE(SCREEN+CELL,TYPRTY)

        OD
        OD
RETURN
```

HALLOWEEN PICTURES FROM THE J.A.C.G. PRINT SHOP DISK

JACK-O-LANTERN

FRANKIE

HAUNTED

SEPTEMBER MEETING HIGHLIGHTS

Reported by
Joseph S. Kennedy

Frank Pazel opened the meeting for the question and answer period where a wide variety of technical and general problems were asked and answered. Again the Q&A session showed that the best people to learn about an Atari from are users.

In Art's opening comments he reported that Ron Luks of Compuserve and Neil Harris of Atari would be at the next meeting. Also at the next meeting there will be nominations for officers and a live demo of an ST. Bob Mulhearn was appointed as the new secretary. Persons are needed to fill the posts of President, Treasurer, Ad Manager and Program Chairman.

Art gave us a slide show of the ST. Sort of a preparation for next month's live demo.

Scott reported that the BBS has been struck by lightning. The equipment is either being repaired or replaced as appropriate and the board will be back up as soon as possible.

Shree VandenBerg gave us a demo of some of the many features of Paper Clip - the word proceesor from Batteries Included. She also explained how she is able to make good use of Paper Clip and her Atari in her accounting business. She might also just be our next Treasurer.

Frank Pazel led a round of applause for Jerry Frese who designed and made our official group banner (which can be seen on stage during the meeting). Frank demoed another international program. This time it was "Herr der Hoehle" (Cavelord) a German maze program that has a flying character similar to Joust. He also showed us Yahtman, a rather good version of the popular game Yahtzee and Maxi-Golf by Adventure International, a golf game that lets you design your own golf course.

Oscar Granison gave us another demonstration of the computer and music.

Bob Mulhearn talked about his modification to make a 256K Atari.

****
GIVE A BIT!!
****

COMING SOON!

SIGN UP FOR YOUR PERSONAL COPY NOW

ATARI COMPUTER CATALOG & BUYERS GUIDE

GEMINI

1986

HARDWARE
SOFTWARE

201-267-0388

86 RIDGEDALE AVENUE
CEDAR KNOLLS, NJ 07927

Forth In The Laboratory

by Donald Forbes - JACG

In today's undeclared submarine war under the oceans (Where do you hide or seek a missile bearing sub in an ocean canyon?), what would be your choice of a disposable computer to run sea-bottom instrumentation?

JACG member Dave Green thought you might be interested in the following highlights from an article on 'FORTH: The optimum language for microcomputers' by Professor Ferren MacIntyre of the Rhode Island University graduate school for oceanography (American Laboratory, Feb-Mar 85). He discusses the software and hardware, backed by an NSF grant, for an ocean-going bubble spectrometer to count and size naturally occurring bubbles in the surface ocean. Rockwell Internationl has packaged a 6502 microprocessor, a Forth kernel, and much additional circuitry in the 40-point R65F11 chip with 16 I/O lines, which forms the heart of the disposable computer to run sea-bottom instrumentation. Here is Dr. MacIntyre's story:

The programming language Forth has, in the author's opinion, been overlooked as a useful laboratory tool. The purpose of this two-part paper is to acquaint readers with applications of Forth, and to suggest that time invested in learning Forth is well spent. Whether one is interested in saving space, minimizing compilation time, or rapid execution, Forth performs notably better than other languages in the microcomputer environment. Most projects spend more time in development than in execution, making fast compilation particularly valuable. The results of a benchmark run by a graduate student innocent of prior knowledge of computers, who is a fair surrogate for the average experimenter, showed that an IBM PC running a bubble sort program that took 2.1 Kbytes and 291 seconds in FORTRAN and 1.1 Kbytes and 36 seconds in Pascal took 700 bytes and 22 seconds in MMS Forth.

Originally, Forth was a FORTRAN program running on an IBM 1130, a 'third generation' computer of the late 1960s. The name was intended to convey the idea of a 'fourth-generation' language, but the 1130 would accept only five-letter names. The program was Charles Moore's response to his employers' penchant for changing mainframes periodically, and his intent was three-fold: to create a way of writing portable applications adaptable to any mainframe with a short machine-dependent kernel; to keep the language simple; and to make it easy to add whatever capability later became desirable.

That Forth turned out to be the ideal language for microcomputers may not be serendipitous, but rather a remarkable case of convergent evolution: good programming evolving independently in the same direction as good machine design. The next step in this evolution is a silicon chip designed to run Forth. Now in the prototype stage, this promises a 20-MHz clock and one Forth primitive per clock cycle -- speeds approaching the CRAY-1 supercomputer and 100 times faster than today's microcomputers.

Forth neatly links hand calculators and supercomputers. I have borrowed programs from both the HP-65 and the CRAY-1, translated then into Forth, and run them on an IBM PC, not as a stunt, but (in what is beginning to seem a characteristic of Forth programming) because this was the most efficient way to complete the desired job.

Have you ever wished that your calculator were not I/O limited? The statistical programs of the HP-65 (Hewlett-Packard) are so useful that I have written them in Forth (the language structure is very similar). The commands and documentation remain intact, but it is possible now to enter data from keyboard, memory, or RS232 line, obtain results 1000 times faster, display them on the screen or printer, save them to disk, or use them in further programs.

Again, have you ever tried accessing a supercomputer from a remote site? The 1000-fold speed advantage of the CRAY over the PC (with the 8087 numerical coprocessor) vanishes rapidly! An IBM task which takes all day in BASIC or 15 minutes in Forth will run in 1 second on the CRAY -- but it typically takes five minutes to get a usable telephone link to the CRAY and tell it what is wanted, reducing its speed advantage to 3:1. In any case, most CRAY jobs are run as background, with 24-hour turnaround, so for results needed today, I use the IBM and Forth.

Structure of Forth

Forth is not so much a language as a programming environment, or set of tools for generating good code. It is usually its own operating system (i.e., it initializes the machines, runs the I/O, formats and copies disks, has a debugger, assembler, editor, etc.). (Versions which run under MSDOS or CP/M may gain in file-handling but also inherit the inadequacies of the system, a trade-off necessitating careful thought.)

Forth uses stack-oriented, reverse-Polish notation (RPN), for the same reason that Hewlett-Packard chose it for calculators: RPN is one of the few ways in which people and computers can gracefully think alike. These structures are familiar from childhood, for RPN is the way we first learned to deal with numerical operations. RPN is much like introductory arithmetic: put a number on the blackboard (or stack), put another number below it, then write the operator, then replace the stack entries with the result. If it now seems strange, try to recapture the first time you saw an algebraic equation, and the feeling of helplessness that it induced! Like most forgotten skills, RPN can be relearned rather easily. The stack is simply a place to store things temporarily, organized exactly like its prototype on a first-in, first-out basis. Access to the top is easy; elsewhere, possible but less easy. It obviates the need for 'local variables,' and is the customary way of passing parameters.

Programming is a way of converting a sequence of logical ideas into a list of machine-executable instructions, or program. The device that does this is a 'compiler' if it produces permanent code, or an 'interpreter' if it is interactive. Forth has both: that is, if the user types 2 3 + .

the interpreter will add the numbers with + ('plus') and display a 5 on the screen with . ('dot'). If the user instead types : TEST 2 3 + ; then : ('colon') turns on the compiler, which adds TEST to the dictionary, and ; ('semicolon') stops compilation. If we now type TEST the interpreter will find TEST in the dictionary and execute it, putting a 5 on the screen.

Traditional compilers struggle with the difference between human and computer thinking; BASIC minimizes the struggle by limiting its vocabulary; and Forth simply takes advantage of the similarities. The results are so striking that it is worth watching how this is done.

The classical way to associate human and machine thinking is through a dictionary, often as a 'vectored' list, in which the human names, or 'headers,' have pointers to their machine-code 'bodies.' A slightly more efficient storing is the 'in-line' dictionary, requiring the same space, but only one growth area. Pointers now link to the preceding entry. Another name for this is 'direct threaded code.' Note that like the original list of machine instructions, there are loops, branch points, and subroutine calls, because a string of instructions without decision structures would be so inflexible that it could not, for example, respond to a keyboard.

The final step is to reduce the machine code to a list of addresses rather than instructions. What this loses in scurrying, it regains in brevity, since most 'subroutines' will be reused. This 'indirect threaded code' is the structure of Forth.

We can now see that the power of Forth lies in its ability to convert ideas directly into programs, maintaining the original structure. Each of the subtasks -- Forth words -- like ACCEPT-INPUT is now decomposed into a set of simpler instructions, until we come to tasks so simple that the machine already knows how to do them.

Because it works with instead of against the programmer, the Forth compiler takes about four lines of Forth code. It and a small dictionary can be put into 1/K or Read Only Memory (ROM), permitting single chip devices to speak a high-level language.

'Writing a program' means adding new words to the dictionary, so that the new 'program' is in no way separated from what has gone before. Everything that the language can do, can be done from the middle of the program: Users may rewrite one sector of a disk, drive I/O ports, read an RS232 line, call upon some feature of the screen editor, dump memory, redefine a printer font, switch monitors, get a screen print, drop into assembler, inquire about plotter status, or perform any other function, provided only that its determining word is in the dictionary.

Lest this total control overwhelm available memory, FORGET <name> removes all words down through <name>. Other words can then be read from disk, providing unlimited virtual memory.

Forth instructions and data words share a common structure. Forth words can be lists of machine instructions, headerless code as in a vectored dictionary, or any number of varieties of data or other structures. In addition, there are defining words, or 'parents,' which create families of 'children' with various names but common behavior. These usually define new data types, so that if a user needs 7-dimensional arrays of pairs of 64-bit-wide complex numbers, there is a way to obtain them. Defining words are more complicated than ordinary words, because they must specify both the compilation-time behavior of the parent when it is creating the child, and the run-time behavior of the child when it is being used.

Pros and cons

Forth is not universally beloved, and some common criticisms include the following: Forth is a 'write-only language,' or a 'language for geniuses,' with a 'small user base' of 'fanatics.' Users of the language C claim they 'will wipe it out.' Forth 'has no floating point math,' and while it may be 'useful for small programs, it is inadequate for large ones.' These criticisms are addressed below.

1) Write-only. Con: Anyone can write unreadable code in a dozen languages, including Forth.

Pro: It is easier to make sense of undocumented Forth than of undocumented BASIC. The usual complaint about Forth is its low redundancy, which is at the root of its differences from other languages. If 'nouns' are data, 'verbs' act upon data, and in FORTRAN, BASIC, and Pascal, the nouns are explicitly interspersed among the verbs. In Forth, the nouns tend to hide on the stack, making the program seem like a string of expletives.

Again, an ordinary language consists of repetitions of control structures, each enclosing a few lines of functional code. The programmer's eye converts the pattern of redundancy into the flow of control. The chief virtue of Pascal is its enforcement of this structure, making it easy for an instructor to detect a 'good' program by eye. In Forth, control structures are distributed one per word, and the unprepared simply cannot find a 'program' without reorganizing their thinking. But good Forth code reads much like (slightly stilted) English.

2) Language for geniuses. Con: Charles Moore may well be a genius, and the users of Forth are at the forefront of their fields (which include astronomy, textiles, primary education, image processing, expert-system creating, robotics, graphics, arcade games, 'Star Wars' special effects, hydrogen fusion, and the manufacture of navigation equipment, plywood, computer aided design tools, microdensitometers, computers, and Fourier-transform spectrometers).

Pro: An elementary class was split, with half learning BASIC first, the other half learning Forth first. Each half was then taught the other language. The responses were prompt and vocal: Halfway through the first session, the BASIC group was indignant that the others were so far ahead. The Forth group, when asked to learn BASIC, saw no point to it and did not.

3) Small user base. Con: Fifteen years ago there were only two Forth programmers.

Pro: Many of today's estimated 30,000 Forth users have come to it reluctantly and in desperation, because nothing else would

work for them.

4) C's industrial base. Con: With both Bell Labs and Digital Equipment Corporation behind it, C is assured of permanent support. Some C users are sure that this means the demise of all competitors.

Pro: C is not an interactive language, and so misses the principal advantage of microcomputers. Being optimized for DEC computers, C is as opaque as assembly language, and produces suboptimal code on other machines. Many professionals develop C and FORTRAN programs in Forth, then translate.

5) Integer arithmetic. Con: In Forth's original laboratory environment, all data were digital integers, and the language incorporates a number of features which optimize the speed and accuracy of integer arithmetic. Purists make use of integers an article of faith, 'in the spirit of Forth.'

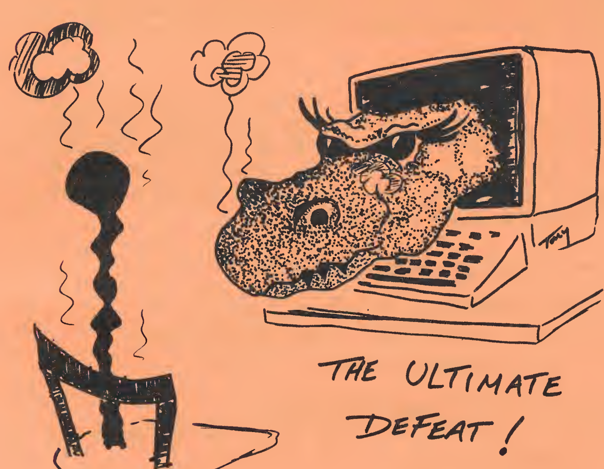
Pro: Moore advocated integers 'until there is hardware floating point' -- meaning the 8087 numerical coprocessor. MMS Forth is an adequate tool for computation-bound floating point number crunching, such as the Mie light scattering equations, which rational people do on supercomputers because they demand recursively calculated Bessel functions, Legendre polynomials, infinite series, complex arithmetic, 16-digit precision, and sophisticated graphics.

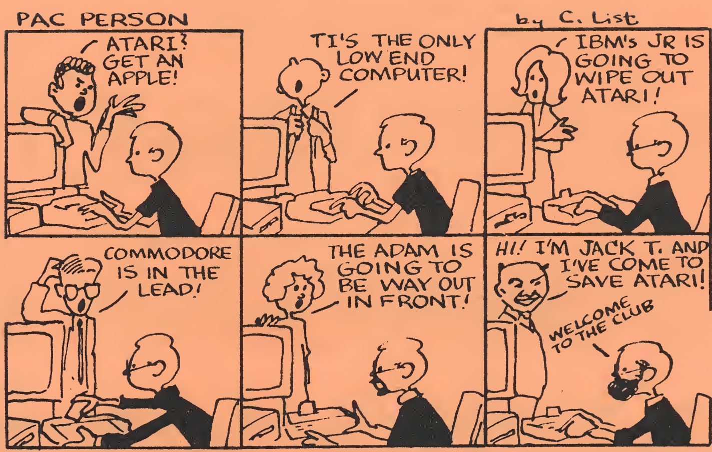
7) Useful only for small applications. Con: The forte of Forth is its ability to put a fast high-level language into the smallest possible space. In turn, it does not need many features customary in languages designed for megaprograms, and does not enforce typographic and stylistic rigidity a la Pascal.

Pro: For the applications envisioned in this article, size is not a stumbling block. However, General Electric's massive locomotive-repair 'expert system' was written in Forth for the usual reason: it was the best way to get the job done.

GIVE A BIT!!!

Contribute to the Newsletter this month.





Portland Atari Club

CHAIN LETTERS

by Kenneth J. Pietrucha - JACG

Every so often it happens. You open your mail and begin reading something which goes like this..."Add you name to the top of this list. Send $10.00 to the name on the bottom of the list and then remove this name. Make ten copies of this letter and sent them to ten friends. When your friends get this letter they will add their names to the top of the list and you will become the number two name. When your name reaches the bottome of the list, the people in position one will send you money ! Joe Mpfhki broke the chain and all his children were born bald."

Now, everyone looks for a quick way to get rich, but before you reach for your wallet, think about this for awhile. I have used an example where you only have to make ten copies, but I know that many chain letters ask you to make 25 copies. For the sake of this discussion, let's stick with my ten copy example.

You send the chain letter to ten people with your name in position one. If the ten people follow the same instructions as you did, then your name goes to position two and there are now one hundred chain letters in circulation with your name on them. When your name reaches position three, there will be 1000 letters. Don't forget, your objective is to reach the bottom of the list which is position ten. If you don't get to position ten, you don't get a dime and you can't quit your job.

Now comes the part I like. As I understand it, there are approximately 80 million households in the United states. If no one breaks the chain by the time your name reaches the eighth position, one hundred million copies of this letter should have been sent, which is 20 million more than there are households. Have you got the message yet ? There are not enough households on the face of the earth to allow you to get your name in position ten.

So, don't quit your job just yet. This is not the way to get rich...besides, it's illegal.

For fun, why not try to write a program to calculate the number of letters in a chain for a different number of copies.

PEEKS AND POKES

by Kenneth J. Pietrucha - JACG

I am always on the look out for new PEEK and POKE locations. The longer you own your ATARI, the more difficult it becomes to find a truly new location.

At the last meeting Frank Pazel passed along a newsleter with some PEEKS and POKES from the Lawrence Atari Computer Club. One of the more interesting locations allows you to modify DOS to let you use lower case letters and punctuation marks in filenames.

To modify DOS, do the following, 1) have a new formatted disk ready, 2)from basic, POKE 3818,33:POKE 3822,123 and 3) go to DOS and use option "H" to write DOS to your new disk.

I tried this and it worked fine. I set up a 2 line dummy file and used lower case letters in the file name. The only problem I had was when I tried to rename the lower case filename with a filename using upper case, I got an error, twice.

In your travels, if you do find a new or odd PEEK and POKE location do pass it along to me.

Until next month...

Harry Brice in Ornjuice
A.C.E. of Orange Co. (CA)

"But Mom, I am taking out the garbage."

J A C G Newsletter Advertisements
RETAILERS - MANUFACTURERS - VENDORS

How would you like to reach a targeted audience of over 700 ATARI computer users? This newsletter has a press run of 800 per month, is read by members and non-members alike, and is sent to over 60 other ATARI User Groups across the U.S. and in several other countries.

Advertising is available on a first-come and space-available basis. Camera ready copy, accompanied by payment, must reach the Editor by the 20th day of the month preceding publication. JACG reserves the right to make decisions concerning the placement of ads within the Newletter and editing or rejecting advertisements deemed unsuitable.

<u>Advertising Rates</u>

Full page (7-1/2 x 9) ... $40.00
Half page $25.00
Quarter page $15.00

Discount rates available upon request.

Contact: Joseph Rowland
429 Washington Street
Hackettstown, NJ 07840
(201) 850-3483

REWARD!!!

****

FREE LIBRARY DISKS FOR GETTING ADS!!!

You can get free disks from our nationally famous and immensely growing JACG Disk Library by simply convincing any retailer, vendor, manufacturer, or whomever, to place an ad in the JACG Newsletter. For every quarter page ad you recruit from someone who has either never advertised in the Newsletter or has not advertised in the last 12 months you will get your choice of one free library disk. Half page ads get you two disks and full page ads get you four disks. How can you beat that deal?

Advertising rates are $15/quarter page, $25/half page, and $40/full page. Your customer's ad reaches a targeted audience of over 600 plus computer store and courtesy mailings which bring the total up to well over 700 per issue. Placement of ads in the Newsletter is the exclusive decision of the Editor but every consideration will be given advertiser requests. Space is available on a first-come, first-served, space-available basis. Contact Advertising Manager Joseph Rowland at (201) 850-3483 for further information or any JACG officer listed on the back page of the Newsletter.

Get out there, talk up the efficacy of advertising in the official JACG publication, drum up some business which will bring down the cost of printing, and earn yourself some disks. This offer is limited to current JACG members and each new advertiser, regardless of repeat business with larger or smaller ads, can be used only once. Members of the Commodore, Apple, and IBM Boards of Directors must apply for special dispensation to qualify for this offer.

===============
JACG Membership
===============

The Jersey Atari Computer Group (JACG) invites you to become a member. Dues are $20.00 per year and entitle the member to: 1) Receive the monthly newsletter; 2) Purchase programs from the group's extensive tape and disk libraries at special rates; 3) Join special interest groups or form new ones; 4) Benefit from the expertise and experience of other Atari computer users; 5) Participate in group purchases of software at substantially reduced prices; 6) Receive a membership card that entites the member todiscounts at local computer stores; 7) Attend monthly meetings to learn about the latest hardware and software, rumors, and techniques for getting the most out of your Atari computer; 8) Submit articles and programs to the newsletter and give demos and presentations at the monthly meetings; 9) Participate in sale/swap activities with other members; 10) Access the JACG nationally famous Bulletin Board; and 11) Have a lot of fun.

If all of this sounds good to you send a check or money order, payable to JACG, to:

Ron Kordos
201 Lake Valley Road
Morristown, NJ 07960

Remember, receiving the JACG Newsletter is just one of the many benefits of being a member of JACG.

DON'T FORGET!
Contribute an article this month.

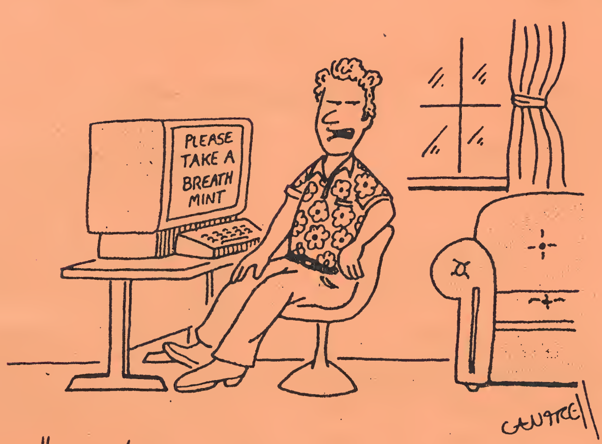
"NAH—YOU WORRY TOO MUCH!"

Portland Atari Club

SOURCES
Hardware-Software-Repair

for your ATARI. Asterisks indicate advertisements elsewhere in this newsletter.

The above listings are free to current advertisers. Others interested in being listed in this column should send business address and telephone number with check for $5 per month, payable to JACG, to advertising manager Joseph Rowland, 429 Washington Street, Hackettstown, NJ 07840.



"GEEZ! TALK ABOUT A PERSONAL COMPUTER!"

FROM MARK CANTRELL!

TRADING POST

Trading Post is a service for JACG members who wish to sell or swap items of any type. There is no charge for this service. Material must reach the Editor by the 20th of the month to be considered for inclusion in the following month's Trading Post. No commercial services or items will be accepted.

>>>>>>>>>><<<<<<<<<<

FOR SALE: KAYPRO 10 Portable Computer with 10 Megabyte Hard disk. Computer is 9 months old and in excellent condition (never had any problems with it). Software includes orginal bundled Wordstar (word processor), Mail Merge (plus Star Index), The Word Plus (spelling checker), DataStar, ReportStar, CalcStar, dBaseII, Microplan, C-Basic, S-Basic, Mite and Automite, selected games. Additional software includes: PromptDoc, T/Maker III, CP/M-86, Turbo Pascal, Pascal 80, SuperCalc 2, Super SpellGuard, Punctuation and Style, AbStat, dUtil, dGraph, Quickcode and about a dozen books. Price? A low $1800 takes the lot. Contact Arthur Leyenberger at (201) 887-2861.

FOR SALE: Commodore 64 computer with 1541 disk drive, printer interface, and over a dozen brand new, high quality programs. Total system - $325. Contact Arthur Leyenberger at (201) 887-2861.

FOR SALE: Atari 1050 printer. $100 or best offer. Contact Ted Colby at (201) 299-2140 days or (201) 366-3655 after 6 p.m.

FOR SALE: ATR8000, 64K w/CPM, (2) 5-1/4" D/S D/D Tandon drives, cabinet and power supply, all cables, manuals and software included, 30 disks, Hayes Smartmodem. Best offer. Call Gerry at (201) 756-3827.

FOR SALE: Panasonic TR-120M green monochrome monitor with sound. Like new. $75.00. Call Marty Henry days at (201) 379-1500.

HAVE YOU RENEWED YOUR MEMBERSHIP?

CHECK YOUR MAILING LABEL
FOR MEMBERSHIP EXPIRATION DATE

###
J A C G
JERSEY ATARI COMPUTER GROUP
40 LAWRENCE ROAD
PARSIPPANY, NEW JERSEY 07054
###

FIRST CLASS MAIL

JACG NEWSLETTER - VOLUME 5, NUMBER 2

October 1985

J A C G EXECUTIVE COMMITTEE

President: Arthur Leyenberger
40 Lawrence Road
Parsippany, NJ 07054
(201) 887-2861 (home)
(201) 386-4254 (work)

Vice Pres.: Scott Brause
12 Bradford Road
Edison,NJ 08817
(201) 549-0878

Secretary: Larry Moriano
80 New Road Apt 30
Parsippany, NJ 07054
(201) 575-8044

Treasurer: Ron Kordos
201 Lake Valley Road
Morristown,NJ 07960
(201) 285-9670

Librarian: Don Ursem
37 Clover Lane
Randolph, NJ 07869
(201) 895-2522

Editor: Frank Pazel
14 Whitman Drive
Denville, NJ 07834
(201) 627-8845

Programs: Ali Chaudry
58 Manor Drive
Basking Ridge, NJ 07920
(201) 647-6822

BBS Sysop: Scott Brause
12 Bradford Road
Edison, NJ 08817
(201) 549-0878

Pres. Emeritus: Richard Kushner
58 Dewey Avenue
High Bridge, NJ 08829
(201) 638-8732

Advertising: Joseph Rowland
429 Washington Street
Hackettstown, NJ 07840
(201) 850-3483

Disk Librarians:
Richard Lamb
Dennis Hoskins
Ken McCollough

The Jersey Atari Computer Group (JACG) is an independent, informal organization of ATARI computer users. It is not affiliated with Atari or any other commercial enterprise. Opinions expressed in this publication reflect only the views of the individual author, and do not necessarily represent the views of JACG. Material in this Newsletter may be reprinted by other Atari User Groups, provided the author (if applicable) and JACG are given credit and the issue date is given. Only original work may be reprinted. Questions concerning reprinting should be addressed to the Editor.